AMENDMENT TO THE CLAIMS

Please **AMEND** claims 1, 3, 24, 26, 36, 43 and 48 as follows.

Please **CANCEL** claims 7-11 without prejudice or disclaimer.

Please **ADD** claims 51 and 52 as follows.

A copy of all pending claims and a status of the claims is provided below.

1. (currently amended) A method of representing and managing rich text for use by Web based applications and browsers as implemented in a machine, the method comprising the steps of:

providing one or more classes for use by the applications to at least create and manage one or more rich text nodes in a memory structure representation representative of rich text;

representing the rich text in the memory structure representation; and

editing the rich text in a document using the memory structure representation to perform editing functions on the document having the rich text as managed and created by the one or more classes, wherein the memory structure representation comprises:

forming a table structure represented in memory as a set of special rich text node types including table node, table body node and table header node for defining table characteristics, wherein the table row node, heading cell node and row cell node correspond to types of html tags controlling table representation, wherein

each type of node maintains a reference to the nodes it controls for a next level including the table row controlling a list of row cell nodes, and the table body node controlling a list of table row nodes,

the header cell node and row cell node maintain lists of the rich text nodes, representing content of the rich text nodes and the rich text node contains an anchor point to another table node to start a new table at that point in the rich text thereby allowing for nested tables; and further comprising:

providing a method to transform text from its memory format into string representations and vice versa, comprising:

storing one of the rich text as a string in a relational database, formatting the string by converting the rich text into html string for storage, converting the rich text into xml and using a compressed format where various attributes of each rich text node are captured, along with the text value for that node, wherein creating rich text memory structure from html, comprises one of:

parsing, by the rich text node, a well-formed segment of html and set its attributes accordingly, including creating other rich text nodes as needed as the html indicates a change in text attributes or presence of an image or link; and

as a function in a rich text list, taking the html that is not well formed, and preprocesses the html to make it recognizable by the rich text nodes, wherein the rich text list also handles creating the nodes for the table structures included within the html.

2. (original) The method of claim 1, wherein the providing the one or more classes includes the steps of:

providing a rich text list class for managing the one or more rich text nodes in the memory structure representation;

providing a rich text class to create the one or more rich text nodes each representing a unit of rich text and its attributes; and

instantiating the rich text list class and the rich text class.

3. (currently amended) The method of claim 1, wherein the representing rich text step includes representing the string representations.

4. (original) The method of claim 3, wherein the string representations comprise at least one of a character large object (CLOB), hyper-text markup language (HTML), extensible markup language (XML), plain text, and spell check text.

5. (original) The method of claim 1, wherein the providing one or more classes step includes providing rich text attributes, wherein the attributes include at least one of font face, font size, font color, italicized, underlined, and bold.

6. (original) The method of claim 1, wherein the providing one or more classes step includes providing properties associated with the one or more rich text nodes, the properties comprising at least one of a line break, a table, an image, a link, and text.

7. –11. (canceled)

12. (previously presented) The method of claim 1, further comprising the steps of:

providing well-formed segments of text to a current rich text node of the one or more rich text nodes from a rich text list node;

parsing the well-formed segments of text;

assigning unparsed segments of text to the current rich text node's text attribute; and

resolving the current rich text node's text attribute by extracting tag in formation and setting attributes in the current rich text node, the attributes including at least one of font face, font size, font color, italicized, underlined, and bold.

13. (original) The method of claim 12, wherein the providing well-formed segments step comprises the steps of:

suppressing certain tags associated with some the unparsed segments by changing starting and ending tags to substitution strings;

checking whether the starting and ending tags are in proper order and eliminating pairs of the starting and the ending tags that have null content;

converting some of the substitution strings to original values; and

reconstituting the well-formed segments of text into one string when pairs of starting and end tags are eliminated.

14. (original) The method of claim 12, wherein the providing well-formed segments step comprises the steps of:

restoring table related tags; and

breaking the well-formed segments at table tags and organizing the broken segments into a new rich text list node with entries of at least one of vectors and string.

15. (original) The method of claim 12, wherein the text is at least one of hypertext mark-up language (html) and extensible mark-up language (xml).

16. (previously presented) A method of representing and managing rich text for use by applications as implemented in a machine, the method comprising the steps of:

providing one or more classes for use by the applications to at least create and manage one or more rich text nodes in a memory structure representation representative of rich text;

representing the rich text in the memory structure representation; and

editing the rich text in a document using the memory structure representation to perform editing functions on the document having the rich text as managed and created by the one or more classes, further comprising the steps of:

providing well-formed segments of text to a current rich text node of the one or more rich text nodes from a rich text list node;

parsing the well-formed segments of text;

assigning unparsed segments of text to the current rich text node's text attribute; and

resolving the current rich text node's text attribute by extracting tag information and setting attributes in the current rich text node, the attributes including at least one of font face, font size, font color, italicized, underlined, and bold;

wherein the resolving step comprises the steps of:

a) reading the text attribute up to a first tag;

b) if the reading step produces a non-null string, then cloning the current rich text node to make a preceding rich text node and assigning to it all text before the tag;

c) checking whether the first tag has a matching end tag;

d) if there is a matching end tag, cloning the current rich text node to make a following rich text node and assigning to it any text after the matching end tag, then removing the text after the matching end tag;

e) resolving the information between the first tag and matching end tag to set up attributes in the current rich text node; and

f) repeating steps a) through e) until a null string is produced in step b).

17. (original) The method of claim 16, further comprising the step of repeating steps a) through f) on one of the preceding rich text node and the following rich text node.

18. (original) The method of claim 16, further comprising the step of when the first tag is one of an image tag and a link tag in step a), cloning the current rich text node to make the following rich text node and assigning to the following node the text after the first tag, then continuing with step c).

19. (original) The method of claim 1, further comprising the steps of:

responding to a request for editing a document containing the rich text;

presenting rich text editing controls for editing the document; and

accepting changes to the document using one or more classes including a rich text class and a rich text list class for editing the document.

20. (original) The method of claim 19, wherein the accepting changes step includes accepting changes to at least one of a table, a link, an image, and text.

21. (original) The method of claim 19, wherein the responding step further comprises steps of:

responding to a spell checking request;

presenting a spell check panel that displays spelling alternatives to a misspelled word associated with the one or more rich text nodes; and

accepting a spelling substitution.

22. (original) The method of claim 21, wherein the responding to a spell checking request step includes searching a spelling dictionary to locate one or more words for presentation in the spell check panel.

23. (original) The method of claim 22, wherein the one or more words in the dictionary each have one or more associated signatures to aid in locating a match for the misspelled word.

24. (currently amended) A method of representing and managing documents having rich text for use by Web based applications and browsers as implemented in a machine, the method comprising the steps of:

representing rich text in a memory structure representation;

providing one or more classes for use by the applications to create the memory structure representation, the one or more classes including a rich text list class to create a rich text list node and to manage one or more rich text nodes and a rich text class to create the one or more rich text nodes each representing a unit of the rich text; and

providing well-formed segments of text to the one or more current rich text nodes from a rich text list node to initialize the current rich text nodes for representing rich text in a document; and further comprising:

providing a method to transform text from its memory format into string representations and vice versa, comprising:

storing one of the rich text as a string in a relational database, formatting the string by converting the rich text into html string for storage, converting the rich text into xml and using a compressed format where various attributes of each rich text node are captured, along with the text value for that node, wherein creating rich text memory structure from html, comprises one of:

parsing, by the rich text node, the well-formed segments of html and set its attributes accordingly, including creating other rich text nodes as needed as the html indicates a change in text attributes or presence of an image or link; and as a function in the rich text list, taking the html that is not well formed, and preprocesses the html to make it recognizable by the rich text nodes, wherein the rich text list also handles creating the nodes for the table structures included within the html.

25. (original) The method of claim 24, further comprising the steps of:
instantiating the rich text list class and the rich text class; and
editing the rich text in the document using the rich text nodes created by the rich text class.

26. (currently amended) The method of claim 24, wherein ~~the representing rich text step includes representing string representations,~~ the string representations ~~including~~ include at least one of ~~a~~ the compressed format, hyper-text markup language (HTML), extensible markup language (XML), plain text, and spell check text.

27. (original) The method of claim 24, wherein the rich text includes attributes of at least one of font face, font size, font color, italicized, underlined, and bold.

28. (original) The method of claim 24 wherein the one or more rich text nodes includes properties, the properties comprising at least one of a line break, a table, an image, a link, and text.

29. (original) The method of claim 24, wherein the one or more rich text node comprises a table node for defining a table and the table node includes at least one of a table header node and a table body node, for defining the characteristics and format of the table.

30. (original) The method of claim 29, wherein the table header node comprises one or more heading cell nodes, each heading cell node defining another rich text node, and wherein the table body node comprises one or more table row nodes for defining an individual row within the table.

31. (original) The method of claim 30, wherein the one or more table row nodes comprise one or more row cell nodes for defining rich text in a cell in the individual row, each of the one or more row cell nodes defining another rich text node.

32. (original) The method of claim 24, wherein the providing one or more classes step further comprises the step of: providing a spell checker class for use by the applications for locating replacement words in the document having rich text.

33. (original) The method of claim 24, wherein the providing well-formed segments step comprises the steps of:

converting some substitution strings to original values;

suppressing certain tags by changing starting and ending tags to substitution strings;

checking whether start and end tags are in proper order and eliminating pairs of start and end tags that have null content; and

reconstituting segments of text into one string when pairs of starting and end tags are eliminated.

34. (original) The method of claim 24, wherein the providing well-formed segments step comprises the steps of: restoring table related tags; and breaking some of the unparsed segments at table tags and organizing the broken segments into a new rich text list node with entries of at least one of vectors and string.

35. (original) The method of claim 24, wherein the providing well-formed segments of text step further comprising the steps of:

parsing the well-formed segments of text;

assigning unparsed segments of text to the current rich text node's text attribute; and

resolving the current rich text node's text attribute by extracting tag information and sets attributes in the current rich text node, the attributes including at least one of font face, font size, font color, italicized, underlined, and bold.

36. (currently amended) ~~The method of claim 35,~~ A method of representing and managing documents having rich text for use in a machine, the method comprising the steps of:

representing rich text in a memory structure representation;

providing one or more classes for use by the applications to create the memory structure representation, the one or more classes including a rich text list class to create a rich text list node and to manage one or more rich text nodes and a rich text class to create the one or more rich text nodes each representing a unit of the rich text; and

providing well-formed segments of text to the one or more current rich text nodes from a rich text list node to initialize the current rich text nodes for representing rich text in a document,

wherein the providing well-formed segments of text step further comprising the steps of:

parsing the well-formed segments of text;

assigning unparsed segments of text to the current rich text node's text attribute; and

resolving the current rich text node's text attribute by extracting tag information and sets attributes in the current rich text node, the attributes including at least one of font face, font size, font color, italicized, underlined, and bold,

wherein the resolving step comprises the steps of:

a) reading the text attribute up to a first tag;

b) if the reading step produces a non-null string, then cloning the current rich text node to make a preceding rich text node and assigning to it all text before the tag;

c) checking whether the first tag has a matching end tag;

d) if there is a matching end tag, cloning the current rich text node to make a following rich text node and assigning to it any text after the matching end tag, then removing the text after the matching end tag;

e) resolving the information between the first tag and matching end tag to set up attributes in the current node; and

f) repeating steps a) through e) until all a null string is produced in step b).

37. (original) The method of claim 36, further comprising the step of repeating steps a) through f) on one of the preceding rich text node and the following rich text node.

38. (original) The method of claim 36, further comprising the step of when the first tag is one of an image tag and a link tag in step a), cloning the current rich text node to make the following rich text node and assigning to the following node the text after the first tag, then continuing with step e).

39. (withdrawn) A method of providing a spellchecker function for use with documents having rich text, the method comprising the steps of:

initializing a dictionary containing words;

creating at least one signature for each dictionary word;

keying the at least one signature to the dictionary word; determining that a word is misspelled by checking the dictionary for the misspelled word resulting in a null value;

creating at least one signature associated with the misspelled word; searching the dictionary using the at least one signature associated with the misspelled word and dictionary word to locate at least one replacement word with the same at least one signature; and

providing the at least one replacement word in the document having rich text.

40. (withdrawn) The method of claim 39, wherein the at least one signature associated with the misspelled word and for each dictionary word is provided by extracting one or more letters and combining the one or more letters.

41. (withdrawn) The method of claim 40, wherein the extracting one or more letters and combining step is provided according to at least one of the following:

a) when the dictionary word or misspelled word is less than three characters, the at least one signature is the dictionary word or misspelled word itself,

b) when the length of each of the dictionary word or misspelled word is greater than eight characters, one signature is the first half of the word,

c) when the length of the dictionary word or misspelled word is eight the first three and last three characters are each signatures,

d) when the length of the dictionary word or misspelled word is between four and seven, the first two characters and last two characters are each signatures,

e) when the length of the dictionary word or misspelled word equals four, the first two characters plus the last character is the signature,

f) when the length of the dictionary word or misspelled word is greater than four, the first four and the last four characters are each signatures, and

g) when the length of the dictionary word or misspelled word equals four, the first character plus the last two characters is a signature.

42. (withdrawn) The method of claim 39, wherein the providing step includes providing more than one replacement words in an ordered list for selection, wherein the more than one replacement words are ordered based upon a score.

43. (currently amended) An apparatus for providing a ~~means~~ mechanism for representing and managing rich text for use by Web based applications and browsers, the apparatus comprising:

a component representing rich text in a memory structure representation;

a component providing one or more classes for use by the Web based applications and browsers to create the memory structure representation, wherein the one or more classes includes,

a) a rich text list class for managing one or more rich text nodes and

b) a rich text class to create one or more rich text nodes each representing a unit of rich text and its attributes,

a component to transform text from its memory format into string representations and vice versa, comprising:

storing one of the rich text as a string in a relational database, formatting the string by converting the rich text into html string for storage, converting the rich text into xml and using a compressed format where various attributes of each rich text node are captured, along with the text value for that node, wherein creating rich text memory structure from html, comprises one of:

parsing, by the rich text node, a well-formed segment of html and set its attributes accordingly, including creating other rich text nodes as needed as the html indicates a change in text attributes or presence of an image or link; and

as a function in the rich text list, taking the html that is not well formed, and preprocesses the html to make it recognizable by the rich text nodes, wherein the rich text list also handles creating the nodes for the table structures included within the html.

44. (original) The apparatus of claim 43, further comprising:
a component instantiating the rich text list class and the rich text class; and
a component editing rich text in a document using the rich text class.

45. (original) The apparatus of claim 43, wherein the component for representing rich text includes representing a string, the string including at least one of a character large object (CLOB), hyper-text markup language (HTML), extensible markup language (XML), plain text, and spell check text.

46. (original) The apparatus of claim 43, further comprising a component for providing spell checking using the memory structure representation.

47. (original) The apparatus of claim 43, wherein the component for representing rich text in a memory structure representation and the component for providing one or more classes for use by the Web based applications and browsers is contained on at least one of a compact disc, a network, a library, a hard drive, a floppy disc, and a memory device.

48. (currently amended) A computer program product comprising a computer usable medium having a computer readable program code embodied in the medium, the computer program product includes:

a first computer program code to provide one or more classes for use by Web based applications and browsers to at least create and manage one or more rich text nodes in a memory structure representation representative of rich text;

a second computer program code to represent the rich text in the memory structure representation; ~~and~~

a third computer program code to edit rich text in a document using the memory structure representation to perform editing functions on the document having rich text as managed and created by the one or more classes; and

at least one further computer program to parse html by extracting tag information from a text attribute, then using the tag information to set other attributes in the one or more rich text nodes by calling a resolveTag method, wherein the resolveTag method comprises:

reading text up to a first tag and if this is not a null string, cloning a current rich text node and making the clone a preceding node, and assigning to it all text before the first tag, then removing the text and calling the resolvetag method again, wherein the html is well formed for the cloning to work recursively, and the well formed html ensures that encountered tags are in proper order so that the text sent to the clone will not miss any tags;

if the tag has a matching end tag, checking if there is any text beyond the end tag, and if there is, cloning the current rich text node, making the clone the following node, and assigning it the text after the end tag, then removing the text and calling the resolveTag method again;

if the tag is an image or link tag, cloning the current rich text node and making the clone the following node, and assigning the following node the text after the tag;

passing the tag information to resolve the tag and to set up tag attributes, wherein if there is an image or link tag, the attributes are stored in the text; and

if preceding or following nodes are not null, call resolveTag making the preceding or following node the current node, which recursively propagates more rich text nodes to fully represent the rich text.

49. (original) The computer program product of claim 48, wherein the computer program product further includes:

a fourth computer program code to provide a rich text list class for creating rich text list nodes and for managing the one or more rich text nodes in the memory structure representation;

a fifth computer program code to provide a rich text class to create the one or more rich text nodes each representing a unit of rich text and its attributes; and

a sixth computer program code to instantiate the rich text list class and the rich text class.

50. (original) The computer program product of claim 49, wherein the computer program product further includes:

a seventh computer program code to provide well-formed segments of text to a current rich text node from a rich text list node;

an eighth computer program code to parse the well-formed segments of text;

a ninth computer program code to assign unparsed segments of text to the current rich text node's text attribute; and

a tenth computer program code to resolve the current rich text node's text attribute by extracting tag information and to set attributes in the current rich text node.

51. (new) The method of claim 1, further comprising:

parsing the html by extracting tag information from a text attribute, then using the tag information to set other attributes in the rich text by calling a resolveTag method, wherein the resolveTag method comprises:

reading text up to a first tag and if this is not a null string, cloning a current rich text node and making the clone a preceding node, and assigning to it all text before the first tag, then removing the text and calling the resolveTag method again, wherein the html is well formed for the cloning to work recursively, and the well formed html ensures that encountered tags are in proper order so that the text sent to the clone will not miss any tags;

if the tag has a matching end tag, checking if there is any text beyond the end tag, and if there is, cloning the current rich text node, making the clone the following node, and assigning it the text after the end tag, then removing the text and calling the resolveTag method again;

if the tag is an image or link tag, cloning the current rich text node and making the clone the following node, and assigning the following node the text after the tag;

passing the tag information to resolve the tag and to set up tag attributes, wherein if there is an image or link tag, the attributes are stored in the text; and

if preceding or following nodes are not null, call resolveTag making the preceding or following node the current node, which recursively propagates more rich text nodes to fully represent the rich text.

52. (new) The method of claim 51, further comprising:

buffering, within each segment html, tags that are not of interest by changing start end and end brackets to substitution strings, which includes a table and list related tags, which are ignored and restored later;

checking to ensure that the tags start and end in the proper order, and each start tag has a matching end tag within the segment, performed by bubbling up end tags that do not have matches within the segment, and then eliminating pairs of start and end tags that have no intervening content;

reconstituting the segments into one string, using a rich text node separator; and

breaking the html into segments at <table> tags, and then organizing the segments into a new rich text list that includes entries that are either simple strings for rich text node entries or vectors for table entries.